The International Journal of Advanced Manufacturing Technology (2019) 103:3613–3627
https://doi.org/10.1007/s00170-019-03739-6

ORIGINAL ARTICLE

Process sequencing for a pick-and-place robot in a real-life flexible robotic cell

Mazyar Ghadiri Nejad[1] · Seyed Mahdi Shavarani[2] · Hüseyin Güden[3] · Reza Vatankhah Barenji[4]

Received: 9 November 2018 / Accepted: 14 April 2019 / Published online: 7 May 2019

Abstract

Robots are used in manufacturing cells for wide purposes including pick and place of the items from a location to a destination. As far as the authors' knowledge in this context, the scheduling problem of a real-life flexible robotic cell (FRC) with intermediate buffers is missing in the literature. Therefore, in this study, the process-sequencing problem of a real-life FRC is considered, aiming to minimize the cyclic operation time of the cell. The problem is mathematically modeled and solved for a real case. Since computation times for solving the problems rise exponentially with increasing the number of machines in the FRC, a genetic, a simulated annealing, and a hybrid genetic algorithms are proposed to solve the large-sized problems. The objective function value of a given solution in metaheuristic algorithms is computed by solving a linear programming model. After tuning the parameters of the proposed algorithms, several numerical instances are solved, and the performance of these algorithms are evaluated and compared. The results show that the performance of the hybrid genetic algorithm was significantly better than both genetic and simulated annealing algorithms.

Keywords Flexible robotic cell · Cyclic scheduling · Manufacturing cell · Metaheuristic

Nomenclature

CNC	Computer numerical control
FRC	Flexible robotic cell
GA	Genetic algorithm
HGA	Hybrid genetic algorithm
MIP	Mixed integer programming
RC	Robotic cell
SA	Simulated annealing
α	The coefficient for temperature modifications
t_a	The completion time of activity a
C	The cycle time
T_f	The final temperature
T_0	The initial temperature
M_i	The ith machine in the FRC
L_{ik}	The loading of the k^{th} item on machine i in each cycle
T^i_{ab}	The lower bound of d_{ab} for machine i
$Iter_{max}$	The maximum number of iterations in GA
N	The number of iterations in SA
m	The number of machines in the FRC
Pop	The number of population
P_c	The probability of crossover
P_m	The probability of mutation
p	The processing time for an item on any machine
w_{ab}	The robot waiting time between t_a and t_b
L_i	The set of loading activities of machine i in each cycle
U_i	The set of unloading activities of machine i in each cycle
ε	The time for just picking/placing an item from/to the input/output buffer, or any machine
d_{ab}	The time of performing activity b after finishing activity a, by the robot
δ	The travel time of the robot between two consecutive station

✉ Reza Vatankhah Barenji
reza.vatankhah@hacettepe.edu.tr

[1] Industrial Engineering Department, Girne American University, Kyrenia, TRNC, Turkey

[2] Alliance Manchester Business School, University of Manchester, Manchester, UK

[3] Department of Industrial Engineering, Eastern Mediterranean University, Famagusta, TRNC, Turkey

[4] Department of industrial engineering, Hacettepe University, Ankara, Turkey

U_{ik} The unloading of the k^{th} item from machine i in each cycle

x_{ab} 1, if the robot performs activity b immediately after activity a; 0, otherwise

z_{ik} 1, if the k^{th} order is applied for the activities of machine i; 0, otherwise

1 Introduction

With traditional issues surrounding the manufacturing industry, there are always opportunities for improvement and industrial robot arms play a significant role in this matter. The use of robots in production has many advantages and benefits such as quality improvement for the system and quality of life improvement for the workers. Robots have the potential to enhance manufacturing sectors from productivity, quality, security, and safety points of view, and can even stimulate the creation of more jobs [1].

Robots in manufacturing cells are mainly used to pick and place the course products/products, assembling, painting, and welding processes. Robotic pick and place speeds up the process and decreases wasted handling time and as a result, increases production rates [2]. With many end-of-arm-tooling options available, robots can be customized to fit specific production requirements and as a consequence, moving large, small, heavy, or hard-to-handle products would be an easy task in manufacturing lines [3]. Consistency is the main benefit of using robots in production [4].

The scope of this research pertains to a real-life flexible robotic cell (FRC). An FRC is comprised of a number of computer numerical control (CNC) machines in which a robot is used to pick, place, and transport the parts between the input/output buffers and machines. Real-life flexible robotic cell is an FRC with intermediate buffers on the CNC machines. In such a cell, each machine is capable of performing all required processes for producing the parts, and item visits only one of the machines in the system [5]. Raw materials are handled by a robot from the main input buffer to intermediate buffers of the machines, and the products should be transported from the machines to the main output buffer. As a result, machines in the cell are laid out in parallel, not a sequence. The flexibility of the machines in the system allows the cell to produce parts with a wide range of features. The robot is capable of transporting parts with different ranges of size and weight. When the number of machines in a cell is high, the robot will be busier. Hence, it is valuable to investigate whether there exists an optimal process sequence for the robot to minimize the idle time of the machines and maximize the productivity of the cell, and if exists, how to find it.

This paper provides an approach to discover the optimal sequence of the pick-and-place jobs in a real-life FRC. The problem is framed as transferring items from the input buffer to the intermediate buffers and from machines to the output buffer. The purpose of this paper is threefold. First, to take into account all the characteristics of the real-life FRC and to develop a mathematical model to solve the optimization problem of the cell. Second, to generalize the problem by considering a cell with m machines and m intermediate buffers. Third, to solve the large-sized problem with simulated annealing (SA), genetic algorithm (GA), and expecting to obtain a much finer result with a hybrid metaheuristic algorithm (the initial parameters of the hybrid metaheuristic algorithms are calibrated by using the Taguchi method). Some numerical examples are solved aiming to make comparisons among the methods.

2 Literature review

According to the industry 4.0 records of the United Nations Educational, Scientific and Cultural Organization (UNESCO), one of the agendas that is perceived as a particular need, and a serious gap in industries is "increase facilities individual flexibility" [6]. By recent developments on technical specifications of CNC machines, robotic cells (RCs) are going to give their place to FRC that will be dominant in the manufacturing systems.

An RC is a group of machines with diverse characteristics in a workspace (e.g., drilling, milling, and turning). These machines are commonly connected to each other by a material handling system in order to process parts [7]. In RCs, the parts are processed through all the machines without any distinction in general. Therefore, parts go through machines in a similar sequence. Each machine performs a single operation and passes the part to the next machine by using the material handling system. A machine in an RC cannot complete all of the operations. In fact, RC scheduling problems lie in classical flow shop, which has been widely studied [8, 9].

In an FRC, all the operations of similar parts with a specific number of different operations are completed on a machine with the same process times [10]. When a part is loaded on a machine, the machine can complete all the remaining operations, not necessarily one operation. Depending on parts operation specifications, there is a need for flexible and programmable machines in the cell to solely perform all of the operations. Hence, it is possible to assign each of the operations to any machine and the operation time is not relative to the machine's name and specifications [11]. The scheduling problem of an FRC can be generalized as two well-known scheduling problems: (a) cyclic scheduling of the robot move, and (b) operation allocation. In the cyclic scheduling of the robot movement, an optimal cycle for the robot's movement sequence among all desired locations in a steady state is considered. On the other hand, in the operation allocation, the optimal order of the operations on the machines is found for each machine, independently. The problem becomes very difficult when the number of machines and operations increases, consequently, existing literature are mostly limited to a simple cell with two

machines for processing a single-part type. The problem even is more complicated by considering constraints for the robot and machines (e.g., considering a different capacity for robot and machines). Moreover, the problem depends on the layout of the cell (i.e., circular or linear configuration).

In the last decades, some researchers have investigated the scheduling problem of FRCs. Akturk et al. [12] suggested a basic framework for FRC scheduling problems. Gultekin et al. [13] considered situations that are more realistic; they divided the operations into three groups based on tool constraints in the tool magazine, and they assumed that the first CNC machine or the second CNC machine could only execute some operations. In other studies, Gultekin et al. assessed the effect of pure cycles on the cycle times [14–16]. Rajapakshe et al. [17] accomplished a comparative productivity analysis on the flexible robotic cell with a single-gripper robot and without swap ability in circular and linear configurations. Yildiz et al. [18] considered a circularly configured FRC with m machines, a shared input/output buffer (I/O), and a single gripper robot without swap ability. Foumani and Jenab [19] investigated an FRC with m machine, a robot with swap ability, and the working machines are allowed to switch parts with each other. Kim et al. [20] examined the cyclic scheduling problem for a dual-armed cluster tool that performs periodic cleaning processes. They identified sufficient conditions for which the conventional backward and swap sequences provide the minimum cycle time. Moreover, they proposed two heuristic scheduling strategies and compared them with traditional programming methods and the lower bound of each schedule. Jiang et al. [21] applied two heuristics to minimize the makespan of a job scheduling problem. They considered a two-machine system where the machines are parallel and identical and the machines are loaded/unloaded by a server. Batur et al. [22] considered a hybrid robotic cell to produce multiple part-types and they presented a TSP-based model to solve this problem. Gultekin et al. [23] studied a two-machine dual-gripper FRC where their aim is finding the best order the robot moves to maximize the production rate. They found five different robot movement orders that dominate the rest of them. Recently, Ghadiri Nejad et al. [5] developed a mathematical model for the scheduling problem of an m machine FRC, and they proposed a SA algorithm for solving the problem. Ghadiri Nejad et al. [24] dealt with the scheduling of a multi-machine robotic cell producing identical parts. Without considering individual buffers for the machines, they tried to maximize the throughput of the system in the long run utilizing a metaheuristic algorithm. Moreover, Nejad et al. [25] considered a bi-objective scheduling problem of a flexible robotic cell to minimize the cyclic production cost of the cell. They proposed a mathematical model and used NSGAII for large-sized problems.

Recently, simulated annealing (SA) and genetic algorithm (GA) solution methods are used to solve a broad range of optimization problems. These methods have come to prominence for their high solution performance and fine results achieved in short times among meta-heuristics approaches. In recent publications, the SA and GA or their modifications have been used for solving the production scheduling problems [26, 27], flow shop FRC scheduling [28], supply chain inventory and network optimization problem [29, 30], emergency logistics problem [31, 32], assembly line balancing problem [33, 34], clustering problem [35, 36], etc.

Observing the cell configuration, the FRC in all the reported studies has a robot, an input/output buffer, and some machines, which are placed on a circular or liner layout. However, a vital component of a real-life FRC, namely an intermediate machine's buffers, is not considered on scheduling problem. As far as the authors' knowledge in this context, the scheduling problem of a real-life FRC with intermediate buffers is missing in the literature.

3 A real-life flexible robotic cell

A real-life FRC with three machines M1, M2, and M3, and one robot is shown in Fig. 1. In this system, the CNC center machines perform milling, drilling, and turning operations on a 19-kg cast crankshaft to be used in an automobile engine. Considering part features, it seems that it is possible to use a flow shop to produce the part. However, because of part sensitivity, the company prefers to do all the machining operations of the crankshaft on a CNC center machine. The part is brought to input buffer by an indexing conveyor. There is an intermediate buffer attached to each of the machines and a robot transfers the parts from the input to the intermediate buffers of the machines, and from machines to the output buffer. Loading of the parts from intermediate buffers to machines is performed by the manipulator of the buffer. A belt conveyor removes the finished parts from the output buffer of the cell. Since the robot performs a repeated sequence of pick-and-place operations, the performance of the cell depends on the sequence of the robot activities. The machining time including the time for loading of each item is 120 s. The load/unload activity by the robot takes 2 s. The transfer velocity of the robot is constant, and all of the individual facilities (i.e., input/output buffers, machines) are located linearly and in equal distance from each other. Therefore, the robot can transfer a part from the input buffer to the first machine, in 5 s which means, the loading of machine 1's buffer from the input buffer, considering the loading and unloading time of the items, takes 9 s for this case. Apparently, the loading of machine 2 and machine 3's buffer from the input buffer takes 14 and 19 s, respectively.

3.1 Problem definition and formulation

In the formulation, m machines are considered in FRC where the processing time for an item on any machine is p. Each machine has an individual input buffer to keep a part. The loading activity of machine i consists of moving the robot to

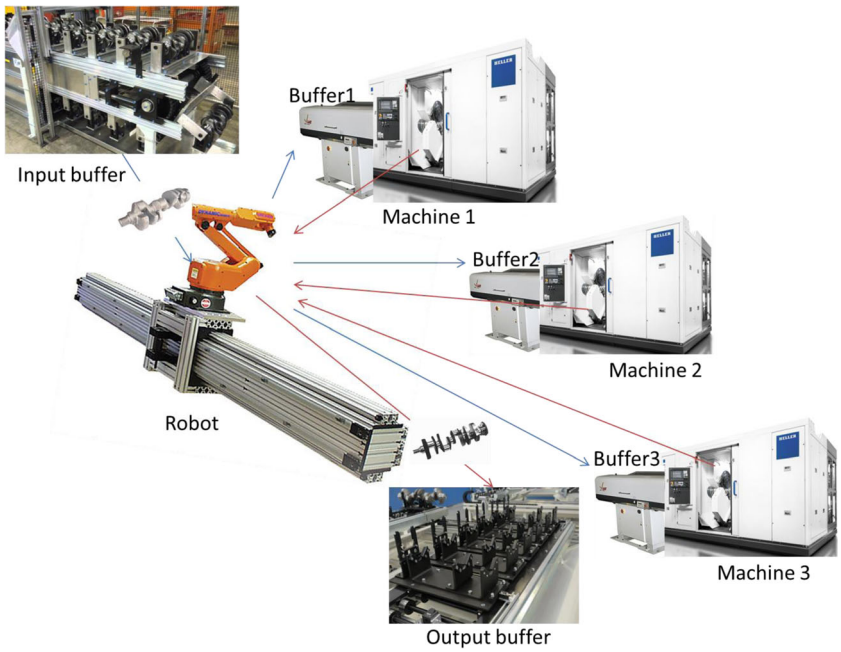

Fig. 1 Real-life FRC configuration for crankshaft production

the input buffer, picking an item, transferring it to machine i, and loading it into the machine's buffer. Similarly, the unloading activity of machine i includes moving the robot towards machine i, getting the finished part from the machine, transferring, and putting it into the output buffer. Note that after each mission, the robot stays in place until its next operation. Cycle time is a duration from the starting of the system from a particular state and returning to the same state.

Let L_{ik} be the loading and U_{ik} be the unloading of the k^{th} item of machine i in each cycle. Let L_i be the set of loading activities of machine i, i.e., $L_i = \{L_{ik} | \ k = 1, 2\}$, and U_i be the set of unloading activities of machine i, i.e., $U_i = \{U_{ik} | \ k = 1, 2\}$, when just two parts can be loaded on each machine. Let A be the set of all loading and unloading activities, i.e., $A = \{L_{ik}, U_{ik} | \ i = 1, 2, \ldots, m; k = 1, 2\}$. Let ε be the time for taking an item from the input buffer or a machine, or the time for putting an item to the machines' buffer or the output buffer. Let δ be the travel time of the robot for a one-unit distance. The distances between the input buffer and the first machine, between any two following machines, and between the last machine and the output buffer are assumed to be one unit of distance. Therefore, the time that the robot needs to perform activity b after finishing activity a (d_{ab}) is:

$$d_{ab} = \begin{cases} 2\varepsilon + (i+j)\delta & \text{if } a \in L_i \text{ and } b \in L_j \\ 2\varepsilon + (|i-j| + m + 1 - j)\delta & \text{if } a \in L_i \text{ and } b \in U_j \\ 2\varepsilon + 2(m + 1 - j)\delta & \text{if } a \in U_i \text{ and } b \in U_j \\ 2\varepsilon + (m + 1 + j)\delta & \text{if } a \in U_i \text{ and } b \in L_j \end{cases}$$

Where m is the number of machines in the FRC. For example, to load machine 3 after loading machine 1 (L_1L_3), the robot goes to the input buffer from machine 1 (1δ), take a part (1ε), goes to machine 3(3δ), and load the machine (ε), that is totally $2\varepsilon + (1 + 3)\delta$ (see the first row of d_{ab}). Until the process of an item has not been finished, the robot may need to wait before unloading it. Therefore, the time between the completion times of activities a and b related to machine i cannot be less than the following values (T^i_{ab}):

$$T^i_{ab} = \begin{cases} 2\varepsilon + (m + 1 - i)\delta + p & \text{if } a = L_{i1} \text{ and } b = U_{i1} \\ 2\varepsilon + (m + 1 - i)\delta + p & \text{if } a = L_{i2} \text{ and } b = U_{i2} \\ \varepsilon + p & \text{if } a, b \in U_i \\ d_{ab} & \text{otherwise} \end{cases}$$

Table 1 contains different scenarios for the states of each machine at the beginning of the cycle.

The order of the loading and unloading activities of each machine can be any of the orders mentioned in this table in the cyclic production. According to the orders of the activities, each machine should be set up before starting the cyclic production, and then the system may repeat the same operations, continuously.

Table 1 All of the activity orders for each machine

The situation of each machine	Activity orders
The machine is idle and its machine buffer is empty	1. $L_{i1}, L_{i2}, U_{i1}, U_{i2}$
	2. $L_{i1}, U_{i1}, L_{i2}, U_{i2}$
The machine has a processed part and its input buffer is empty	3. $L_{i2}, U_{i1}, U_{i2}, L_{i1}$
	4. $L_{i2}, U_{i1}, L_{i1}, U_{i2}$
	5. $U_{i1}, L_{i2}, U_{i2}, L_{i1}$
	6. $U_{i1}, L_{i2}, L_{i1}, U_{i2}$
The machine has a processed part and its input buffer is full	7. $U_{i1}, U_{i2}, L_{i1}, L_{i2}$
	8. $U_{i1}, L_{i1}, U_{i2}, L_{i2}$

Considering the cyclic form of the operations, and for the sake of simplicity and prevention of repeating similar permutations, one activity may be fixed as the first activity. In this study, L_{11} is considered as the first activity. Therefore, the time between two consecutive L_{11} activities is regarded as the cycle time. A reduction in the cycle time means an increase in the production rate of such a system. In the following, a mixed integer programming (MIP) model is proposed to find the optimal cycle time, the order of activities, and the robot waiting time before unloading each machine. The decision variables of this MIP model are as follows: C is the cycle time, t_a is the completion time of activity $a \in A$, and w_{ab} is the waiting time of the robot between the completion times of activity a and following activity b, $a \in A$, $b \in A - a$.

$$x_{ab} = \begin{cases} 1 & \text{if robot performs activity } b \text{ immediately after activity } a \\ 0 & \text{otherwise} \end{cases}$$

$$z_{ik} = \begin{cases} 1 & \text{if } k^{th} \text{ order is applied for the actvities of machine } i; k = 1, 2, ..., 8 \\ 0 & \text{otherwise} \end{cases}$$

$$\min \sum_{a \in A} \sum_{b \in A-a} d_{ab} x_{ab} + \sum_{a \in A} \sum_{b \in A-a} w_{ab} \qquad s.t. \tag{1}$$

$$\sum_{b \in A-a} x_{ab} = 1 \quad \forall a \in A \tag{2}$$

$$\sum_{a \in A-b} x_{ab} = 1 \; \forall b \in A \tag{3}$$

$$t_b \geq t_a + d_{ab} + w_{ab} - M(1 - x_{ab}) \quad \forall b \in A - L_{11}, a \in A - b \tag{4}$$

$$t_b \leq t_a + d_{ab} + w_{ab} + M(1 - x_{ab}) \quad \forall b \in A - L_{11}, a \in A - b \tag{5}$$

$$C \geq t_a + d_{aL_{11}} + w_{aL_{11}} - M(1 - x_{aL_{11}}) \quad \forall a \in A - L_{11} \tag{6}$$

$$C \leq t_a + d_{aL_{11}} + w_{aL_{11}} + M(1 - x_{aL_{11}}) \quad \forall a \in A - L_{11} \tag{7}$$

$$w_{ab} \leq M x_{ab} \quad \forall a \in A, b \in A - a \tag{8}$$

$$\sum_{j=1}^{8} z_{ij} = 1 \quad i = 1, ..., m \tag{9}$$

$$t_{Li1} - t_{Ui1} \leq M(z_{i3} + z_{i4} + z_{i5} + z_{i6} + z_{i7} + z_{i8}) \quad i = 1, ..., m \tag{10}$$

$$t_{Li1} - t_{Ui2} \leq M(z_{i3} + z_{i5} + z_{i7}) \quad i = 1, ..., m \tag{11}$$

$$t_{Li1} - t_{Li2} \leq M(z_{i3} + z_{i4} + z_{i5} + z_{i6}) \quad i = 1, ..., m \tag{12}$$

$$t_{Ui1} - t_{Li1} \leq M(z_{i1} + z_{i2}) \quad i = 1, ..., m \tag{13}$$

$$t_{Ui1} - t_{Li2} \leq M(z_{i1} + z_{i3} + z_{i4}) \quad i = 1, ..., m \tag{14}$$

$$t_{Li2} - t_{Ui1} \leq M(z_{i2} + z_{i5} + z_{i6} + z_{i7} + z_{i8}) \quad i = 1, ..., m \tag{15}$$

$$t_{Li2} - t_{Ui2} \leq M(z_{i7} + z_{i8}) \quad i = 1, ..., m \tag{16}$$

$$t_{Li2} - t_{Li1} \leq M(z_{i1} + z_{i2} + z_{i7} + z_{i8}) \quad i = 1, ..., m \tag{17}$$

$$t_{Ui2} - t_{Li1} \leq M(z_{i1} + z_{i2} + z_{i4} + z_{i6} + z_{i8}) \quad i = 1, ..., m \tag{18}$$

$$t_{Ui2} - t_{Li2} \leq M(z_{i1} + z_{i2} + z_{i3} + z_{i4} + z_{i5} + z_{i6}) \quad i = 1, ..., m \tag{19}$$

$$t_{Ui1} \geq t_{Li1} + T^i_{L1,U1} - M(1 - z_{ik}) \quad i = 1, ..., m; k = 1, 2 \tag{20}$$

$$t_{Ui2} \geq t_{Li2} + T^i_{L2,U2} - M(1 - z_{ik}) \quad i = 1, ..., m; k = 1, 2, 3, 4, 5, 6 \tag{21}$$

$$t_{Ui2} \geq t_{Ui1} + T^i_{U1,U2} - M(1 - z_{ik}) \quad i = 1, ..., m; k = 1, 3, 4, 7, 8 \tag{22}$$

$$t_{Li1} - t_{Ui1} \leq C - T^i_{L1,U1} z_{ik} \quad i = 1, ..., m; k = 3, 4, 5, 6, 7, 8 \tag{23}$$

$$t_{Li2} - t_{Ui2} \leq C - T^i_{L2,U2} z_{ik} \quad i = 1, ..., m; k = 7, 8 \tag{24}$$

$$t_{Ui2} - t_{Ui1} \leq C - T^i_{U2,U1} z_{ik} \quad i = 1, ..., m; k = 4, 6, 8 \tag{25}$$

$$t_a \geq 0 \quad \forall a \in A \tag{26}$$

$$C \geq 0 \tag{27}$$

$$w_{ab} \geq 0 \quad \forall a \in A, b \in A - a \tag{28}$$

$$x_{ab} \in \{0, 1\} \quad \forall a \in A, b \in A - a \tag{29}$$

$$z_{ik} \in \{0, 1\} \quad i = 1, ..., m; k = 1, ..., 8 \tag{30}$$

The objective function of this model is to minimize the cycle time including the robot waiting times. Constraint (2) guarantees that the robot passes from each activity to another activity. Constraint (3) ensures that the robot passes each machine once. Constraints (2) and (3) together guarantee that the robot performs each activity. Constraints (4) and (5) compute the completion times of the activities considering the robot moving time, its loading and unloading times, and the waiting times of the robot to perform successive activities. These constraints also eliminate sub-cycles. Constraints (6) and (7) compute the cycle time considering the completion time of the last activity, the robot moving time, the load/unload times, and the waiting time until completing the first activity of the cycle. If a and b are not successive activities, w_{ab} will be fixed at zero by constraint (8). Constraint (9) guarantees that for each machine, one of the eight possible orders is applied. Constraints (10)–(19) compare the completion times of the activities of each machine and force the related z variables to be 1. Since in all of the eight orders, U_{i1} is before U_{i2}; considering constraint (9), there is no need to use the similar restrictions for U_{i2}–U_{i1}. Constraints (20)–(25) compute the completion times of the activities regarding the processing times of the machines and the time that the robot needs to perform these tasks. Constraints (26)–(28) are the non-negativity constraints, and finally, constraints (29) and (30) define the binary variables.

4 Solution methods

To solve the problem using the mathematical model, enumeration method is used considering the real data of the cell. Genetic and simulated annealing algorithms and a hybrid genetic algorithm are proposed to solve the large-sized cells.

4.1 Genetic algorithm

Genetic algorithm (GA) starts by generating a population of random solutions. Each solution is represented by an array

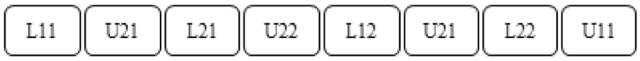

Fig. 2 GA solution representation

having *4m* elements in the form of *XYZ* where *X*, *Y*, and *Z* show loading/unloading activity, machine number, and part number, respectively. As an example, L_{21} illustrates a loading activity on machine 2 for its first part. Figure 2 depicts an encoding of a solution for a four-machine FRC. To avoid having similar permutations of a solution, the loading of the first machine with the first part is fixed as the first activity.

The value of the fitness function for each solution is calculated considering the described formulations and given parameters. Since there is a probability of generating an infeasible solution, a repair procedure as shown in Fig. 4 is applied. It should be mentioned that after each crossover, or mutation operations, the solution is checked and is repaired if it is infeasible. In each step, some offsprings are produced by mating the selected parents through crossover and mutation operators. The one-point-crossover operator is utilized to decompose parent solutions into two segments. The second part of the parents after the crossover point is interchanged and two offsprings/children are produced. A graphical representation of the crossover operation is shown in Fig. 3.

If there exists a repeated gene in the offspring, its second appearance in the chromosome would be replaced by a gene which is absent in the solution. If there are more than one repeated genes, they are replaced by the absent ones in random order. For instance, L_{22} has been repeated twice in the first chromosome of offspring solution in Fig. 3. To repair this chromosome, the first L_{22} remains in the same place, but the second one is replaced by U_{21} which was absent in the solution. Similarly, in the second chromosome of the offspring solution in Fig. 3, U_{21} has been repeated twice and L_{22} which is the only absent gene is replaced in the second repeated gene of U_{21}. Figure 4 illustrates the repaired offspring solutions.

The mutation operator uses shift, swap, and reverse operations to generate neighboring solutions from the current solution. Shift operator selects an activity randomly and changes its place, preserving the order of all the other activities. Swap operator selects two activities randomly and changes their place. The reverse operator selects two positions and reverses the order of activities between them. Figure 5 gives examples for each of these mechanisms.

After generating the offsprings and adding them to the population, they are sorted based on their fitness function values. The best solutions are selected for starting the next iteration. The algorithm continues for a pre-determined number of iterations. Algorithm 1, represents the pseudo code of the proposed GA.

```
Initialize the GA parameter (Pop, Iter_max, P_c and P_m)
Generate initial solution from the size of Pop
Check their feasibility and correct them if needed
Compute the fitness function value for the initial solutions
Iteration_no ⟵ 0
while (Iteration_no < Iter_max) do
    for i = 1: (P_c * Pop)/2
        Select two parents from the population randomly
        Find a crossover point randomly
        Generate two new solutions
        Check their feasibility and correct them if needed
        Compute their fitness function value
        Add them to the population
    End
    for j = 1: P_m * Pop
        Select one solution from the population randomly
        Choose one of the shift, swap and reverse operator randomly
        Perform the mutation
        Check their feasibility and correct them if needed
        Compute its fitness function value
        Add it to the population
    End
    Sort the solutions based on their fitness function values, increasingly
    Select the first Pop size solutions
    Update the population
    Iteration_no ⟵ Iteration_no + 1
end while
Return the first solution;
```

4.2 Simulated annealing

The simulated annealing (SA) algorithm starts by generating a random initial solution at a temperature of $T = T_0$, and its fitness function is calculated similarly to what is explained for GA. SA generates a new neighboring solution by applying one of the shift, swap, and reverse operators on the initial solution, and then the fitness value of the new solution is calculated. If the fitness value is improved, the new solution is accepted for generating the next solution. Otherwise, the new solution with worse fitness value is accepted with a probability calculated by Eq. (31). Having the chance of selecting the worse solution is the advantage of the SA algorithm to escape from the local optimal and find better solutions. After each iteration of SA, the temperature drops by a coefficient of α ($\alpha \in [0, 1]$). The algorithm goes on until the temperature is less than a final temperature (T_f) and the best solution found so far is returned. Algorithm 2 represents the pseudo code of the proposed SA.

$$\mathrm{Exp}(-(OBF(\text{new solution}) - OBF(\text{current solution}))/T) \tag{31}$$

```
1:  Initialize the SA parameter (T0, Tf, α, and N)
2:  Generate an initial solution
3:  Check the feasibility and correct it if needed
4:  Compute the fitness function value for the initial solution
5:  optimal value, current value ⟵ fitness function value
6:  optimal solution, current solution ⟵ fitness function value
7:  T ⟵ T0
8:  while (T < Tf) do
9:      for i = 1: N
10:         Select one of the shift, swap and reverse operator randomly
11:         Generate a new solution from the current solution
12:         Check the feasibility and correct it if needed
13:         Compute its fitness function value
14:         if fitness function value of new solution < Optimal value
15:             optimal value ⟵ new solution value
16:             optimal solution ⟵ new solution
17:         end
18:         if fitness function value of new solution < current value
19:             current value ⟵ new solution value
20:             current solution ⟵ new solution
21:         elseif rand(0,1) < equation (31)
22:             current value ⟵ new solution value
23:             current solution ⟵ new solution
24:         end
25:     end
26:     T ⟵ T * α
27: end while
28: Return optimal value;
```

4.3 Hybrid genetic algorithm

The proposed hybrid algorithm proceeds the same as the GA except it uses a simulated annealing algorithm to improve the quality of each offspring, generated by the crossover operation. If the fitness function value is improved, the offspring is replaced by the new solution. To consider a probability to keep a worse solution for the next iteration, Eq. (32) is used. Algorithm 3 represents the pseudocode of the mutation algorithm which is located instead of lines 17 to 21 of the pseudocode of the GA in Algorithm 1

$$\mathrm{Exp}(-(OBF(\mathrm{new})-OBF(\mathrm{current}))/(1-\mathrm{Iteration\_no}/\mathrm{Itermax})) \quad (32)$$

```
1:  Select one solution from the new offsprings, randomly
2:  for i = 1: N
3:      Select one of the shift, swap and reverse operator randomly
4:      Generate a new solution from the current solution
5:      Check the feasibility and correct it if needed
6:      Compute its fitness function value
7:      if fitness function value of new solution < current value
8:          current value ⟵ new solution value
9:          current solution ⟵ new solution
10:     elseif rand(0,1) < equation (32)
11:         current value ⟵ new solution value
12:         current solution ⟵ new solution
13:     end
14: end
```

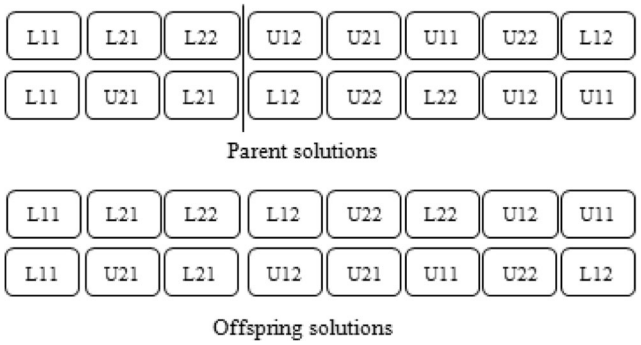

Fig. 3 An example of the crossover operation

5 Experimental results

The experiments of all the problem instances have been performed on a platform with Intel(R) Core(TM) i5-3320 CPU at 2.60GHz and a 4.0 GB RAM.

5.1 The initial results for the real-life FRC case

The real-life FRC as an instance of the proposed model has been solved by an enumeration method, which helps to consider all different robot sequences to complete the jobs. Moreover, its corresponding cycle time allows verifying if all the logic and connections have been properly set. Moreover, for each run, detail information regarding the entire elements of the model and processes carried out in the system were provided.

To analyze the behavior of the real-life FRC, it is interesting to consider the data regarding total robot's cycle time, the sequence of the robot operations, utilization of machines, and robot idle times. The minimum cycle time required to produce two parts in each of the individual machines in the existing configuration of the cell, as shown in Fig. 6, was 288 s. Considering the sequence of the robot operations for the optimal cyclic time, it was possible to conclude that the suggested sequential operations of the robot were logical and properly arranged. Figure 6 demonstrates the operations' Gantt chart of the tasks in the cell for the optimum result.

According to the top row of Fig. 6, the robot performed $L_{21}U_{12}U_{32}U_{22}L_{32}L_{22}L_{12}U_{11}U_{31}U_{21}L_{31}L_{11}$ activities, consequently. This means the cycle was started just after completion of the loading of the first part on the first machine (at time of zero). Then, the robot went to the main input buffer, took a part, went to the second machine and loaded the buffer of the machine (at the time of 19). Then, the robot went to the first machine and unloaded the part, transferred it to the main output

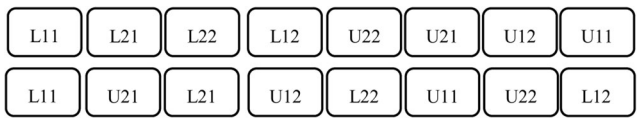

Fig. 4 Repairing the chromosome after a crossover operation

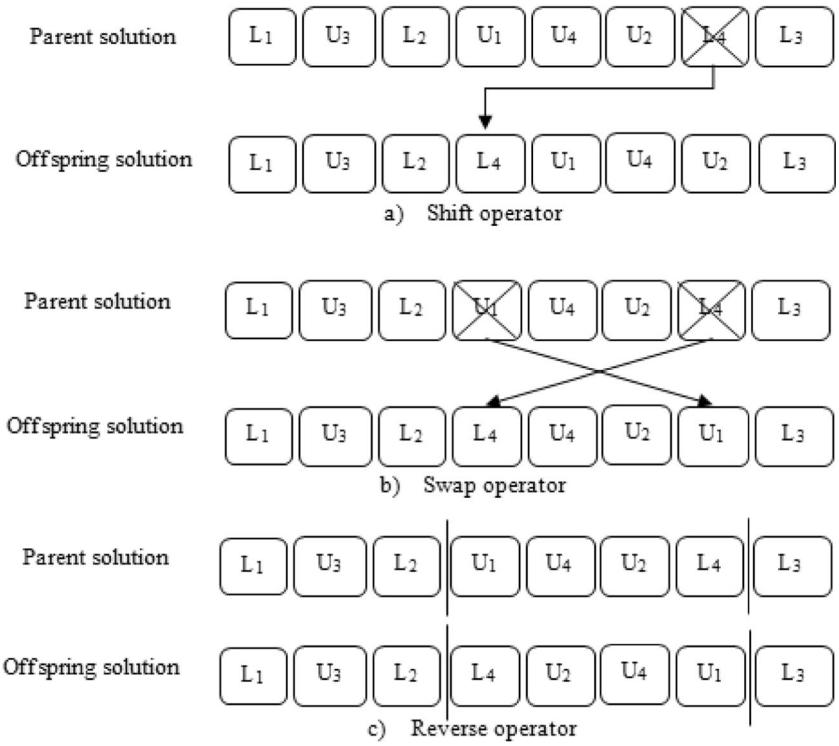

Fig. 5 Generation of neighboring solutions by mutation operator

buffer and put it there (at time 43) and so on. The other three rows of the figure show the processing schedules of the machine where it is seen that when the machines were processing and when they were idle or were waiting to be loaded/unloaded.

In the real-life case, daily production in the cell was divided into three shifts, each consisting of 8 h, for 6 days a week, and consequently, 312 days a year, which means the available time was 26,956,800 s in a year. During this available time, the cell could complete 93,600 cycles, in each of which six parts were produced. Therefore, the cell was able to produce 561,600 number of products each year. Considering the utilization as a percentage of busy time over total production time, the use of the robot was 100%, as it was always busy according to the optimal schedule, and the usage of the machines was similar for all and equal to 83.3%.

Now, let us consider the order of robot activities before finding the optimal one. The previous order was $L_{12}L_{21}L_{22}L_{31}L_{32}U_{11}U_{21}U_{31}U_{12}U_{22}U_{32}L_{11}$. The cycle time of this order was 366 s. It shows that having this order in the real-life case, and considering 26,956,800 s available time in a year, the cell could have 73,652 complete cycle which was able to produce 441,912 parts in a year. Just these data show that the productivity of the cell with the optimized order was about 12.7% more than the cell with the previous order.

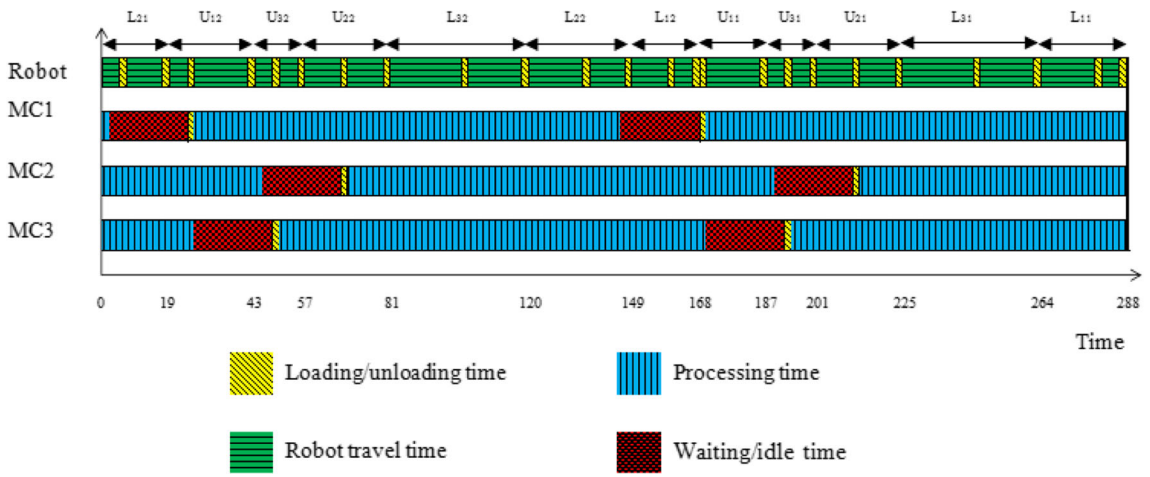

Fig. 6 Gantt chart of the optimal solution of $m = 3$, $\varepsilon = 2$, $\delta = 5$, and $p = 120$ problem

To evaluate the performance of the proposed mathematical model, some instances for a two- and three-machine cells with different processing times were considered. The ε and δ were fixed at one and two time units, respectively. The model was coded in CPLEX 12.6 software. Table 2 illustrates the related results where P was the processing times of the machines, m shows the number of machines in the cell, and C was the optimal cycle time calculated by the model.

As shown in Table 2, the optimal cycle times for both two- and three-machine cells were constant when the processing times of the machines were small enough. Then by raising the processing times, their cycle times were steadily increasing. The computation times for the two-machine problems were always less than 1 s. For three-machine problems, the computation times were enormous when the processing times were small. By increasing the processing times until about 40 units, the computation time climbed to a minute. The other important point was the high rising in computation times between two- and three-machine cell problems with the same processing times, especially when the processing times were small. The experience gained from the model solution for the cell allowed us to draw some conclusions about the proposed model. It was verified that the model worked as specified in both mathematical processes and real-life operation. Additionally, the computation time of the model for large-size cells increased exponentially. It seemed that the use of metaheuristic algorithms for solving the model might be a good candidate.

Table 2 The results of the mathematical models for two- and three-machine problems

p	$m = 2$		$m = 3$	
	C	Computation time (sec.)	C	Computation time (sec.)
0	64	0.76	120	4875.20
5	64	0.67	120	2847.28
10	64	0.57	120	2028.11
15	64	0.43	120	991.49
20	64	0.36	120	492.69
25	64	0.31	120	256.56
30	64	0.34	120	214.78
35	72	0.39	120	82.46
40	82	0.36	120	50.36
45	92	0.35	120	15.32
50	102	0.35	120	12.10
55	112	0.34	120	19.05
60	122	0.37	122	25.12
65	132	0.34	132	17.05
70	142	0.43	142	24.80
75	152	0.35	152	17.09
80	162	0.37	162	20.28
85	172	0.32	172	16.57
90	182	0.37	182	18.68
95	192	0.37	192	15.82
100	202	0.35	202	19.78

Table 3 Levels of the parameters

Algorithms	Parameters	Level (1)	Level (2)	Level (3)	Level (4)
GA	(A) Pop	25	50	75	100
	(B) IterMax	40	50	75	100
	(C) P_c	0.3	0.5	0.7	0.9
	(D) P_m	0.3	0.5	0.7	0.9
SA	(A) T_0	90	95	98	99
	(B) T_f	0.5	0.6	0.75	1.0
	(C) α	0.7	0.8	0.9	0.95
	(D) N	10	15	20	30
HGA	(A) Pop	25	50	75	100
	(B) IterMax	40	50	75	100
	(C) P_c	0.3	0.5	0.7	0.9
	(D) P_m	0.3	0.5	0.7	0.9
	(E) N	5	10	15	20

5.2 Performances of the proposed metaheuristic algorithms

5.2.1 Tuning of the initial parameters

The performance of the proposed GA, SA, and HGA depend on their initial parameter settings. Therefore, different levels for the parameters of the proposed algorithms were selected and illustrated in Table 3.

To find the best initial amounts of the parameters, the Taguchi approach was used [37]. Using Eq. (31), the deviation of the response was examined, wherein Y designated the value of reply and n characterizes the number of orthogonal ranges.

$$S/_N = -10 \times \log10\left(\mathrm{sum}\left(Y^2\right)/n\right) \tag{33}$$

The best configuration of the initial values of the parameters is the one that increases the signal to noise ratio (S/N). For this reason, the S/N ration was plotted for each parameter and the parameters which maximized this ratio were selected, which means the parameters that decrease the effect of noise in the experiments. Seven different problems were considered to find the best combination of the initial values for each algorithm (as shown in Table 4).

Table 4 Problems which are solved to find the best initial parameters

Number of machines (m)	4	5	6	7	8	9	10
Processing times (P)	90	140	200	240	290	330	380

Table 5 The results of different parameter combinations for the proposed algorithms

Order	GA parameters				Fitness function value	SA parameters				Fitness function value	HGA parameters					Fitness function value
	A	B	C	D		A	B	C	D		A	B	C	D	E	
1	1	1	1	1	619.86	1	1	1	1	774.00	1	1	1	1	1	608.43
2	1	2	2	2	571.71	1	2	2	2	650.29	1	2	2	2	2	545.86
3	1	3	3	3	548.86	1	3	3	3	563.00	1	3	3	3	3	538.86
4	1	4	4	4	546.57	1	4	4	4	558.43	1	4	4	4	4	539.57
5	2	1	2	3	581.14	2	1	2	3	622.00	2	1	2	3	4	545.00
6	2	2	1	4	562.57	2	2	1	4	645.71	2	2	1	4	3	543.14
7	2	3	4	1	552.00	2	3	4	1	590.86	2	3	4	1	2	539.00
8	2	4	3	2	549.64	2	4	3	2	598.86	2	4	3	2	1	538.86
9	3	1	3	4	555.29	3	1	3	4	572.43	3	1	3	4	2	547.00
10	3	2	4	3	554.14	3	2	4	3	559.50	3	2	4	3	1	542.29
11	3	3	1	2	550.43	3	3	1	2	704.93	3	3	1	2	4	538.86
12	3	4	2	1	538.86	3	4	2	1	746.50	3	4	2	1	3	538.86
13	4	1	4	2	545.86	4	1	4	2	553.14	4	1	4	2	3	543.71
14	4	2	3	1	553.50	4	2	3	1	639.07	4	2	3	1	4	538.86
15	4	3	2	4	547.14	4	3	2	4	598.14	4	3	2	4	1	538.86
16	4	4	1	3	541.00	4	4	1	3	748.00	4	4	1	3	2	539.00

For each combination of the parameters, the problems were solved five times, and the average of the fitness values was used as the response in Taguchi analysis. This method was deployed by Minitab 16.0. The design of the experiments can be found in Table 5.

Figure 7 illustrates the S/N ratios for GA. According to this figure, all of the GA parameters were better to be at their fourth level. In other words, GA performed better when the initial value of population size was set to its upper bound (100), the maximum iterations was 100, and the probability of crossover and mutation were the same and equal to 90%.

The results of the Taguchi analysis for SA were depicted in Fig. 8. It was evident that T_0, T_f, α, and N should be in their second, third, fourth, and fourth levels which correspond to 95, 0.75, 0.95, and 30, respectively. Additionally, according to Fig. 9, the parameters of HGA including Pop, $Iter_{Max}$, Pc, Pm, and N were better to be set at 100, 75, 0.7, 0.7, and 20, respectively.

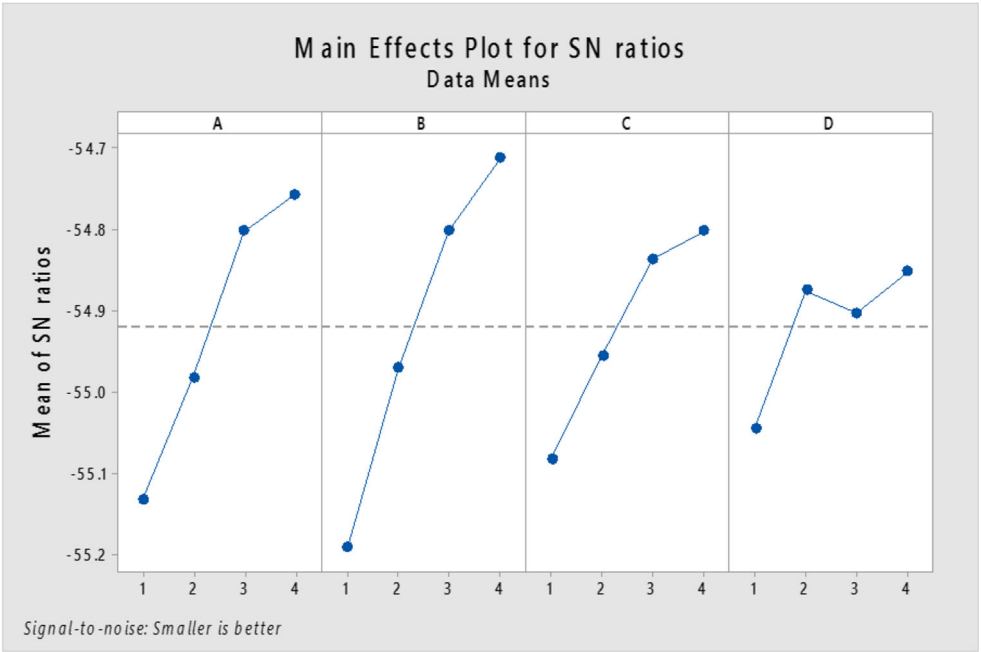

Fig. 7 S/N ratio plot for initial parameters of GA

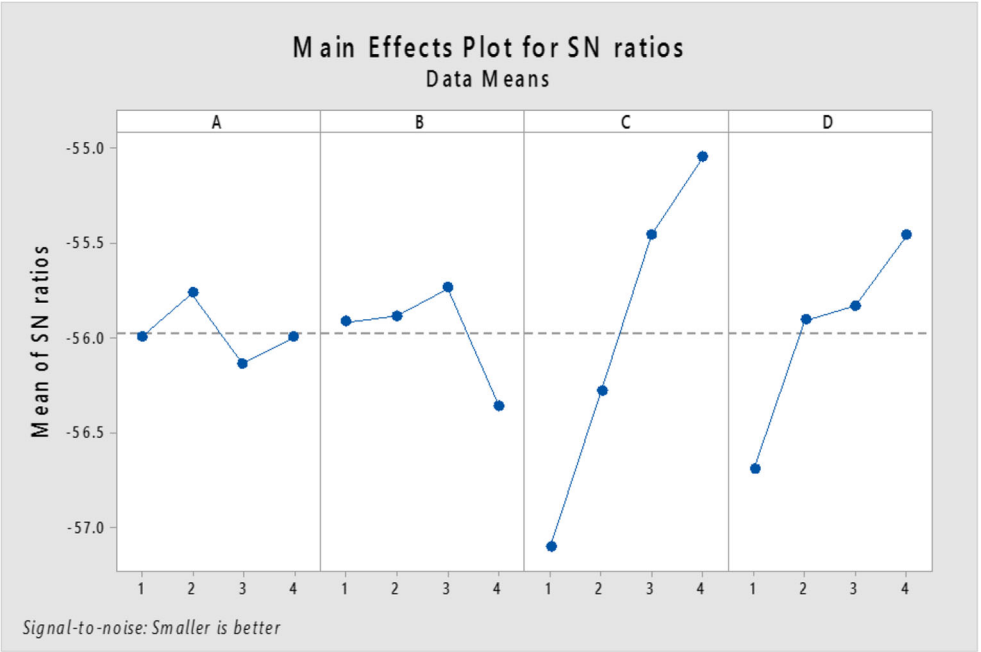

Fig. 8 S/N ratio plot for initial parameters of SA

5.2.2 Comparisons between performances of the developed algorithms

For performance comparison of the algorithms, their parameters were set at their optimum values found by the Taguchi method, and some instance problems with four-, six- and eight-machine cell with various processing times were solved ten times each. The averages of the results were reported in Table 6.

Considering the results shown in Table 6, both GA and HGA have got similar or better results than the SA for all three different cell problems. Moreover, the comparison between the results of GA and HGA shows that the HGA found the same or better solutions than the GA algorithm.

Table 7 shows the computation times for obtaining the results in Table 6. To have a better comparison among computation times, the results of each problem instance with the same number of the machine in the cell was plotted separately. Figures 10, 11, and 12 are related to four-, six- and eight-machine cell problems, respectively. These figures have shown similar trends for the

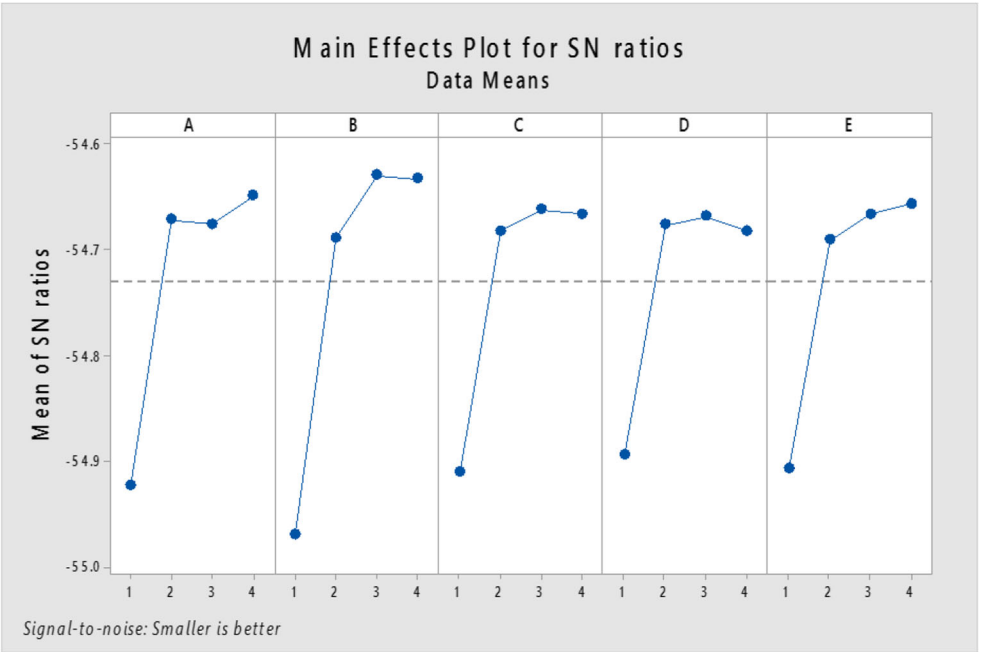

Fig. 9 S/N ratio plot for initial parameters of HGA

Table 6 The cycle times found by the proposed algorithms

Process time	4-machine cell			6-machine cell			8-machine cell		
	GA	SA	HGA	GA	SA	HGA	GA	SA	HGA
0	192	192	192	384	384	384	640	640.8	640
50	192	192	192	384	384	384	640	640	640
100	202.3	204	202	384	384.4	384	640	640	640
150	302	304.3	302	384	384	384	640	640.4	640
200	402	404.3	402	405.1	414.35	402	640	642.3	640
250	502	502	502	502	510.4	502	640	645.3	640
300	602	606.6	602	602	614.2	602	641.8	670.6	640
350	702	704.3	702	702	710.6	702	702	739.4	702
400	802	802	802	802	819.8	802	805.9	852.8	802

Table 7 The computation times of the proposed algorithms

Process time	4-machine cell			6-machine cell			8-machine cell		
	GA	SA	HGA	GA	SA	HGA	GA	SA	HGA
0	2.9	1.7	4.3	4.1	6.2	1.9	7.3	14.0	13.4
50	13.4	10.7	14.0	11.9	8.9	13.4	15.4	13.5	18.1
100	395.9	33.4	755.8	32.7	16.5	34.4	29.7	12.8	42.3
150	382.3	32.7	735.2	75.7	25.3	135.7	48.7	21.5	70.3
200	365.6	31.7	726.6	415.6	32.1	791.7	94.2	27.9	153.4
250	367.0	31.0	658.0	403.5	32.2	784.7	169.7	32.3	275.9
300	367.7	34.5	651.8	391.4	32.9	771.5	379.5	35.3	559.7
350	355.6	36.1	646.9	386.5	34.4	744.9	474.6	33.8	859.1
400	338.2	36.2	655.1	392.7	34.6	744.4	454.8	36.3	878.0

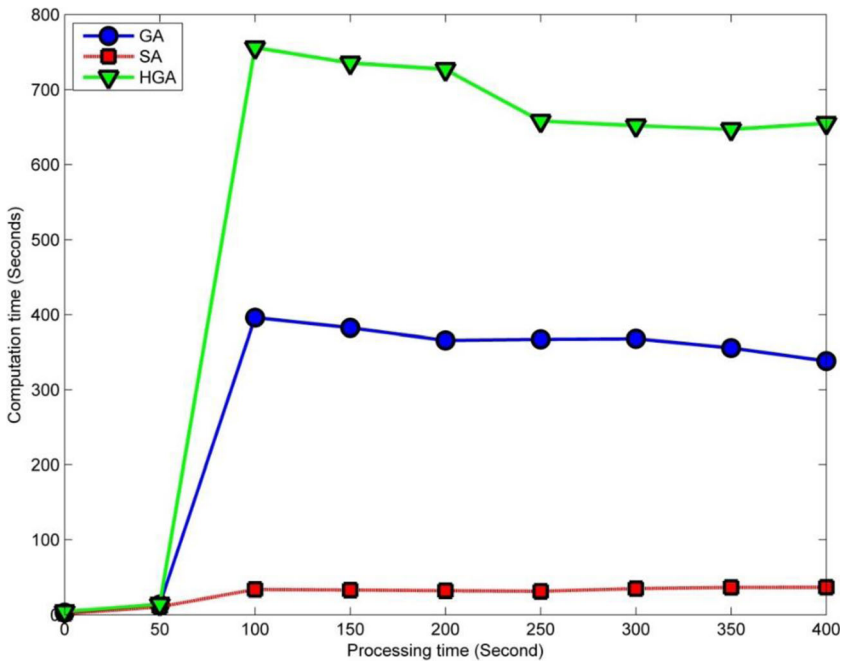

Fig. 10 Computation times of the proposed algorithms for four-machine test problems

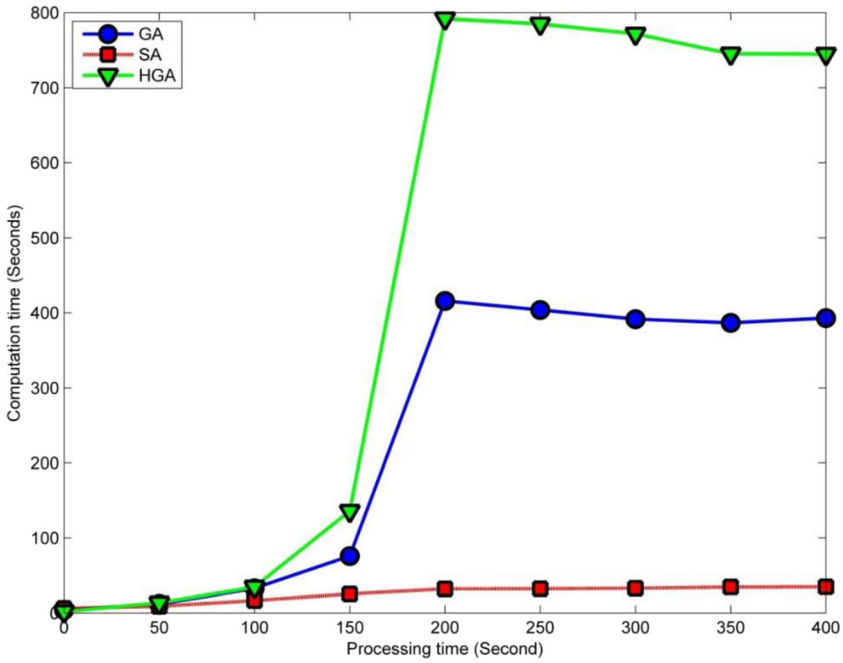

Fig. 11 Computation times of the proposed algorithms for six-machine test problems

computation times of the algorithms. It is apparent that the solution time of the SA was significantly lower than the other two algorithms and it remained constant by increasing the machine processing times in all the solved problems. On the other hand, the computation times of the GA and HGA had similar trends. For small processing times, the computation times of both algorithms were small, then by increasing the processing times, after a sharp rise, they had decreased a little bit and reached to a steady state situation. It should be mentioned that the computation time of the HGA was always greater and it increased sooner and sharper than the GA, except for the eight-machine cell problems, where the computation time of GA raised before the HGA.

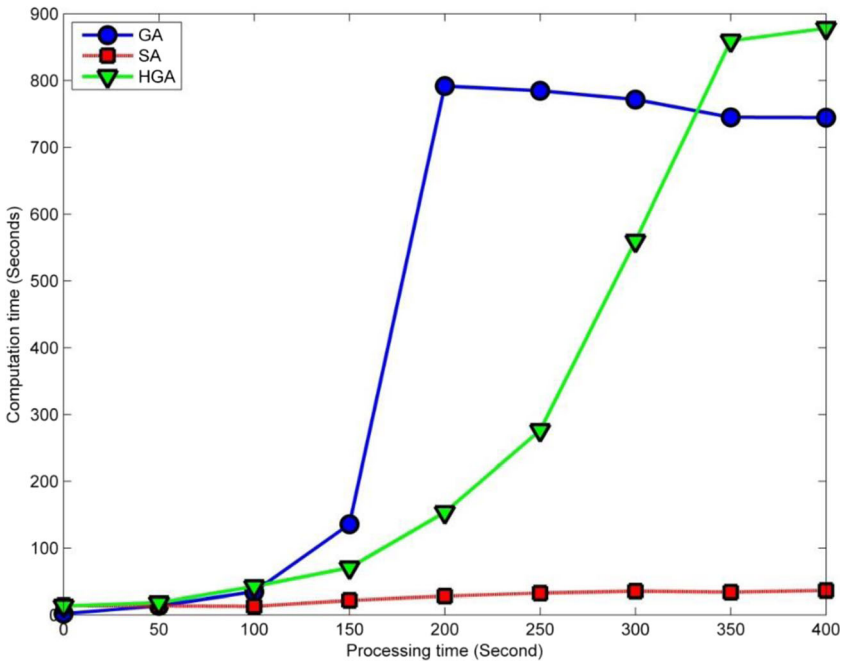

Fig. 12 Computation times of the proposed algorithms for eight-machine test problems

6 Conclusion

In this study, the process sequencing problem of a pick-and-place robot in the real-life flexible robotic cell (FRC) was addressed. The problem aimed to minimize the cell's cycle time by finding the best sequence of the operations for the robot to perform the jobs. The problem was solved by developing a mixed integer programming model for a real-life FRC. Moreover, by addressing some two- and three-machine cell scenarios, it was indicated that the computation times for solving the problems rose exponentially. Therefore genetic, simulated annealing, and a hybrid genetic algorithm were proposed to solve the large-size problems. Furthermore, utilizing the Taguchi method, the best combinations of initial parameters for the algorithms have been set. The comparisons of the results gained by the proposed algorithms illustrated that the performance of the proposed hybrid genetic algorithm dominates the other algorithms of this study.

Proposing a new mathematical model for the given problem with different capacities of the individual input buffer can be an attractive objective for further researches. The case that the machines have both individual input and output buffers may also be considered for future studies. Additionally, considering these problems under uncertainty of each parameter can be the next subject of discussion.

References

1. Barenji RV, Barenji AV, Hashemipour M (2014) A multi-agent RFID-enabled distributed control system for a flexible manufacturing shop. Int J Adv Manuf Technol 71(9–12):1773–1791
2. Barenji RV (2013, September) Towards a capability-based decision support system for a manufacturing shop. In: Working conference on virtual enterprises. Springer, Berlin, Heidelberg, pp 220–227
3. Vatankhah Barenji R, Hashemipour M, Guerra-Zubiaga DA (2015) A framework for modelling enterprise competencies: from theory to practice in enterprise architecture. Int J Comput Integr Manuf 28(8): 791–810
4. Barenji RV, Hashemipour M, Guerra-Zubiaga DA (2013, April) Toward a modeling framework for organizational competency. In: Doctoral conference on computing, electrical and industrial systems. Springer, Berlin, Heidelberg, pp 142–151
5. Ghadiri Nejad M, Güden H, Vizvári B, Vatankhah Barenji R (2018) A mathematical model and simulated annealing algorithm for solving the cyclic scheduling problem of a flexible robotic cell. Adv Mech Eng 10(1):1–12
6. Erol S, Jäger A, Hold P, Ott K, Sihn W (2016) Tangible industry 4.0: a scenario-based approach to learning for the future of production. Procedia CiRp 54:13–18
7. Gultekin H, Tula A, Akturk MS (2016) Automated robotic assembly line design with unavailability periods and tool changes. Eur J Ind Eng 10(4):499–526
8. Lei W, Che A, Chu C (2014) Optimal cyclic scheduling of a robotic flowshop with multiple part types and flexible processing times. Eur J Ind Eng 8(2):143–167
9. Huang KL, Ventura JA (2013) Two-machine flow shop scheduling with synchronous material movement. Int J Plann Sched 1(4):301–315
10. Foumani M, Smith-Miles K, Gunawan I (2017) Scheduling of two-machine robotic rework cells: in-process, post-process and in-line inspection scenarios. Robot Auton Syst 91:210–225
11. Vizvari B, Guden H, Ghadiri Nejad M (2019) Local search based meta-heuristic algorithms for optimizing the cyclic flexible manufacturing cell problem. Ann Optim Theory and Pract 1(3): 15–32
12. Akturk MS, Gultekin H, Karasan OE (2005) Robotic cell scheduling with operational flexibility. Discret Appl Math 145(3):334–348
13. Gultekin H, Akturk MS, Karasan OE (2006) Cyclic scheduling of a 2-machine robotic cell with tooling constraints. Eur J Oper Res 174(2):777–796
14. Gultekin H, Akturk MS, Karasan OE (2007) Scheduling in a three-machine robotic flexible manufacturing cell. Comput Oper Res 34(8):2463–2477
15. Gultekin H, Akturk MS, Karasan OE (2008) Scheduling in robotic cells: process flexibility and cell layout. Int J Prod Res 46(8):2105–2121
16. Gultekin H, Karasan OE, Akturk MS (2009) Pure cycles in flexible robotic cells. Comput Oper Res 36(2):329–343
17. Rajapakshe T, Dawande M, Sriskandarajah C (2011) Quantifying the impact of layout on productivity: an analysis from robotic-cell manufacturing. Oper Res 59(2):440–454
18. Yildiz S, Karasan OE, Akturk MS (2012) An analysis of cyclic scheduling problems in robot centered cells. Comput Oper Res 39(6):1290–1299
19. Foumani M, Jenab K (2012) Cycle time analysis in reentrant robotic cells with swap ability. Int J Prod Res 50(22):6372–6387
20. Kim H, Kim HJ, Lee JH, Lee TE (2013) Scheduling dual-armed cluster tools with cleaning processes. Int J Prod Res 51(12):3671–3687
21. Jiang Y, Zhang Q, Hu J, Dong J, Ji M (2015) Single-server parallel-machine scheduling with loading and unloading times. J Comb Optim 30(2):201–213
22. Batur GD, Erol S, Karasan OE (2016) Robot move sequence determining and multiple part-type scheduling in hybrid flexible flow shop robotic cells. Comput Ind Eng 100:72–87
23. Gultekin H, Dalgıç ÖO, Akturk MS (2017) Pure cycles in two-machine dual-gripper robotic cells. Robot Comput Integr Manuf 48:121–131
24. Nejad MG, Kovács G, Vizvári B, Barenji RV (2018) An optimization model for cyclic scheduling problem in flexible robotic cells. Int J Adv Manuf Technol 95(9–12):3863–3873
25. Nejad MG, Shavarani SM, Vizvári B, Barenji RV (2018) Trade-off between process scheduling and production cost in cyclic flexible robotic cells. Int J Adv Manuf Technol 96(1–4):1081–1091
26. Behjat S, Salmasi N (2017) Total completion time minimisation of no-wait flowshop group scheduling problem with sequence dependent setup times. Eur J Ind Eng 11(1):22–48
27. Nestic S, Stefanovic M, Djordjevic A, Arsovski S, Tadic D (2015) A model of the assessment and optimisation of production process quality using the fuzzy sets and genetic algorithm approach. Eur J Ind Eng 9(1):77–99
28. Mosallaeipour S, Nejad MG, Shavarani SM, Nazerian R (2018) Mobile robot scheduling for cycle time optimization in flow-shop cells, a case study. Prod Eng 12(1):83–94
29. Liu Q, Xu J, Zhang Z (2015) Construction supply chain-based dynamic optimisation for the purchasing and inventory in a large scale construction project. Eur J Ind Eng 9(6):839–865
30. Jiang D, Li H, Yang T, Li D (2016) Genetic algorithm for inventory positioning problem with general acyclic supply chain networks. Eur J Ind Eng 10(3):367–384

31. Shavarani SM, Nejad MG, Rismanchian F, Izbirak G (2018) Application of hierarchical facility location problem for optimization of a drone delivery system: a case study of Amazon prime air in the city of San Francisco. Int J Adv Manuf Technol 95(9–12): 3141–3153
32. Ghadiri Nejad M, Banar M (2018) Emergency response time minimization by incorporating ground and aerial transportation. Ann Optim Theory Pract 1(1):43–57
33. Güden H, Meral S (2016) An adaptive simulated annealing algorithm-based approach for assembly line balancing and a real-life case study. Int J Adv Manuf Technol 84(5–8):1539–1559
34. Nejad MG, Kashan AH, Shavarani SM (2018) A novel competitive hybrid approach based on grouping evolution strategy algorithm for solving U-shaped assembly line balancing problems. Prod Eng 12(5):555–566
35. Al-Araidah O, Dalalah D, Azeez MEAA, Khasawneh MT (2017) A heuristic for clustering and picking small items considering safe reach of the order picker. Eur J Ind Eng 11(2):256–269
36. Lee MC, Wee HM, Wu S, Wang CE, Chung RL (2015) A bi-level inventory replenishment strategy using clustering genetic algorithm. Eur J Ind Eng 9(6):774–793
37. Vatankhah Barenji R, Ghadiri Nejad M, Asghari I (2018) Optimally sized design of a wind/photovoltaic/fuel cell off-grid hybrid energy system by modified-gray wolf optimization algorithm. Energy Environ 29(6):1053–1070